# DECUS
## PROGRAM LIBRARY

| | |
|---|---|
| DECUS NO. | 8-250 |
| TITLE | FAST FOURIER TRANSFORM (FFT) |
| AUTHOR | Kenneth G. Pavel |
| COMPANY | Trinity College<br>Hartford, Connecticut |
| DATE | February 1970 |
| SOURCE LANGUAGE | PAL-D |

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

# FAST FOURIER TRANSFORM (FFT)

DECUS Program Library Write-up DECUS No. 8-250

## OPERATING INSTRUCTIONS

Starting Address = 0200

Program immediately halts at location 0200. Set the switch register for desired mode of operation.

Switch settings

SWO up----Convert and sample on RC timing clock flag.
SWO down- Convert and sample every 0.01 seconds.

N many samples are taken and then displayed on the scope. If

SW11 up---Keep displaying input time series until lowered.
SW11 down- Display only once.

Compute the frequency spectrum of the input, in less than 8 seconds, and display the output on the scope.

SW10 up----Keep displaying until lowered.
SW10 down-Display once then return to beginning of the program.
SW9 up-----Display histogram.
SW9 down --Display only points.
Knob 34----Controls 8 step vertical scaling.
Knob 35---Controls 32 step horizontal scaling.
Knob 36----Controls 1024 step starting frequency value.

The Fast Fourier Transform

Very often it is necessary to know more about a signal than just what it or n-many averaged out look like. For purposes of analysis, it is often helpful to know what the frequency components of the waveform are. A good example of this is the analysis of electroencephalograms (EEG's) to determine sleep stages. Determining the frequency components of a wave involves a shift from the time domain into the frequency domain. This can be done using Fourier analysis.

The underlying assumption of the whole field of Fourier analysis is that every piecewise continuous, absolutely integrable function can be approximated by an infinite series of sine and cosine waves, or;

$$f(t) = \sum_{n=0}^{n=\infty} (a_n \cos n w_0 t + b_n \sin n w_0 t) \qquad \text{(equ. 1.)}$$

where; $$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(nw_0 t)\, dt \qquad n=0,1,2,\ldots$$

and $$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(nw_0 t)\, dt \qquad n=1,2,3,\ldots$$

In which T is the period of the function, and $w_0$, and $w_0=2/T$.

By noting the relationship between the trigonometric functions and the complex exponentials;

$$e^{jx} = \cos(x) + j\sin(x)$$

$$\sin(wt) = \frac{1}{2j} \left(e^{jwt} - e^{-jwt}\right)$$

$$\cos(wt) = \frac{1}{2} \left(e^{jwt} + e^{-jwt}\right) \qquad \text{(equ. 2.)}$$

where; $j = \sqrt{-1}$

and $e = 2.72$ is the base for natural logarithms.

the Fourier series (equ. 1. ) becomes;

$$f(t) = \sum_{n=0}^{n=\infty} c_n e^{jnw_0 t} \qquad \text{(equ. 3.)}$$

where; $$c_n = 1/2\ (a_n - b_n) = \left| c_n \right| e^{j\phi_n}$$

$$\left| c_n \right| = 1/2(a_n^{\,2} + b_n^{\,2})^{1/2}$$

$$\phi_n = \arctan(-b_n/a_n).$$

If we go to the limiting cases, and let T approach infinity, the Fourier series becomes the Fourier integral. This integration process is known as the Fourier Transform and is defined as;

$$f(w) = F(f(t)) = \int_{-\infty}^{\infty} f(t)\, e^{-jwt}\, dt \qquad \text{(equ. 4.).}$$

Equation 4, as was mentioned above, was for piecewise continuous functions. However, if digital analysis techniques are to be used in the analysis of the waveform, then it is necessary that the data be sampled in order to produce a time series of discrete samples which can be fed into a digital computer. The fundamental theorem of sampling theory says that if a waveform is frequency band limited and the samples are taken at a rate at least twice that of the highest frequency present, then the time series completely and accurately represents the continuous waveform. These samples need not be equally spaced in time, but for the purposes of analysis, and simplicity, they will be considered as if they were equally spaced. The frequency twice the highest present in the waveform is known as the Nyquist or aliasing frequency. The Fourier transform that is taken of this time series is known as the Discrete Fourier Transform (DFT).

The formation of the time series may be looked upon as the multiplication of a continuous function, x(t), by a sampling function, s(t). In this case, s(t) is actually an infinite series of impulses. Mathematically, if two functions are multiplied, then their Fourier transforms are subjected to a process called convolution. This makes the frequency function, $F(x(t)) = x(w)$, a periodic function, where each period contains complete information about the frequency components of the original continuous waveform. This frequency spectrum has a period equal to the inverse of the spacing between samples. Because of this periodicity, only one period need be calculated. Graphical representation of this is provided in figure 1.

The frequency components of the DFT are defined by equ. 5.

$$S(n) = \Delta t \sum_{k=0}^{k=N-1} X(k)\, e^{-i2\pi(nk)/N} \qquad n=0,1,2,\ldots,N-1$$

(equ. 5.)

Where X(k) is the $K^{th}$ time sample, and N is the total number of samples taken.

If T is taken to be the time interval between the samples, then the true frequency is given by equ. 6.

$$f = k/(NT) \qquad \text{(equ. 6.)}$$

Thus it is very important to know both N and T if the interpretation of S(n) is to be meaningful.

Equation 5. is often written in the form shown by;

$$S(n) = \Delta t \sum_{k=0}^{k=N-1} X_k W^{rk} \qquad r=0,1,2,\ldots,N-1$$

(equ. 7)

where; $W = e^{-i2\pi/N}$.

Evaluation of equ. 7. would seem to imply $N^2$ complex multiplications. However, Cooley and Tukey devised a method of evaluating it if N is an integer power of 2. This process involves no more than $N\log_2 N$ complex multiplications. Because of the vast reduction in computation time, this algorithm became known as the Fast Fourier Transform (FFT). Computationally, equ. 7. is nothing more than a complex matrix multiplication of the form;

$$\left[S(n)\right] = \left[W^{rk}\right]\left[X_o(k)\right] \qquad \text{(equ. 8.)}$$

where; $\left[S(n)\right]$ and $\left[X_o(k)\right]$ are N *1 column matrices,

and; $\left[W^{rk}\right]$ is an N*N square matrix.

The term $\Delta t$ is only a scalar multiplier and is of little computational value, and thus, ignored. It should also be noted that the elements of $\left[S(n)\right]$ will be in binary bit reverse order if $\left[X_o(k)\right]$ is properly ordered, and vice versa. We see that since all the elements of the matrices are complex, a matrix multiply routine would take up too much space and too much time. So the implementation of the FFT will be discussed.

Start by drawing a vertical array of N ($=2^g$) nodes. This array is called $X^0$, so that sample $X_k$ is associated with the $K^{th}$ node of $X^0$. The number k will be expressed in binary form, and will take on values ranging from 0 to N-1. Now construct g many of these arrays to the right of the original. Each node has a number within it, and a number associated with it (but not inside it). The number $X_k^o$ is the number associated with the $k^{th}$ node in the $e^{th}$ array. (e is now just an index, and not to be confused with the natural logarithm base.)

Looking at figure 2., a tree-graph, every node has a solid and a dashed line drawn to it from two different nodes in the previous array. The solid lines stand for multiplication, and the dashed ones stand for addition, the two processes that are involved in a complex multiply. If the number in the $k^{th}$ node of the $e^{th}$ array is C, and the solid line is drawn to that node from node A, and the dashed line comes from node B, then $X_k^e$ is formed by taking the number associated with A and multiplying it by C, and then added to the number associated with B. Letting lower case letters a, b represent the numbers associated with nodes A, B, and process is described by equ. 9.

$$X_k^e = aW^C + b \qquad \text{(equ. 9.)}$$

The number within the $k^{th}$ node of the $e^{th}$ array is also formed in a simple manner. Since k is a binary number of g bits, scale it (g-e) many places to the right, filling the leftmost bits with 0's as they are vacated. After scaling has taken place, reverse the order of the bits. This reversal, or inversion, produces the number in the $k^{th}$ node of the $e^{th}$ array.

The rules for locating the arrows of the $k^{th}$ node in the $e^{th}$ array are as follows: The solid line to the $k^{th}$ node in the $e^{th}$ array comes from the node in the $e\text{-}1^{st}$ array whose address (vertical position) is the same as k's except bit $k_{g-e}$ is set to 1. The dashed line originates at the node in the $e\text{-}1^{st}$ array whose address is the same but $k_{g-e}$ is set to 0.

By using the above rules, a tree-graph for any $N=2^g$ inputs can be drawn. The numbers associated with the nodes in the second array are computed from the inputs, using the arrow rules. Similarly, the numbers associated with the nodes in the third array come from the numbers associated with the nodes of the second array, and so on. The numbers associated with the nodes of the last array are actually the S(n)'s of equ. 7.

Computationally, it is not necessary to create new arrays continually, rather, the computations can be done in place. This allows for a considerable saving of space, and a larger number of

input values. It should also be noted that the values of $W^k$ can either be calculated on the spot, or found in a look-up table, which allows for a further saving of time. Furthermore, the creation of the second and third arrays, the first and second passes, can be reduced from complex multiplications to simple additions and subtractions because of the nature and value of $W^k$. This is enhanced if the input data is real, with no imaginary values.

The program that was written to perform the FFT takes samples using the analog to digital converter of the AX08. The sampling rate can be either fixed or variable. Display of the input time series is provided. The output is also in the form of a display. The final smoothed frequency spectrum can be viewed through a variable sized window, which is under the control of the parameter knobs on the front panel. The Spectrum that is displayed is a magnitude only spectrum, and modified in that the components $a_n$ and $jb_n$ are only squared and added; the square root is not taken, and the $\Delta t$ term is left out. For matters of scaling, the multiplication routine has been normalized so that when a number is multiplied by the largest number available in the computer, the answer is twice the initial. Single precision is used throughout, and in order to take care of fractional frequencies, and to provide statistical validity, the data was subjected to Hanning processes twice. The Hanning process weighs the coefficient with the sum of the weighed elements. The program is designed to handle any $N=2^9$ inputs as long as the data is stored with the beginning of the imaginaries N away from the reals. Further restrictions are that N must be less than 1024 ($=2^{10}$), and there must be a trigonometric table for values of sine equal in length to the number of data points.

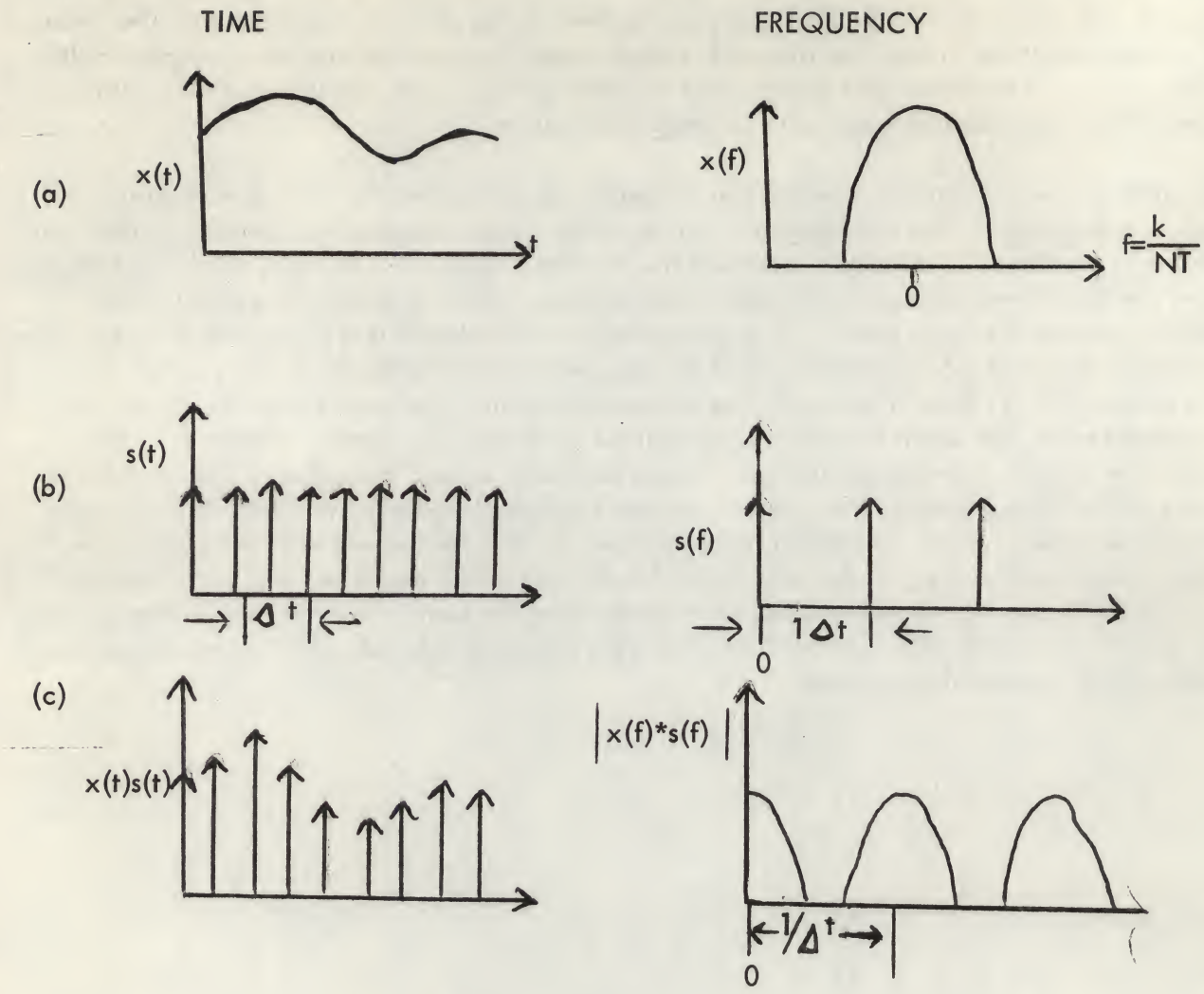


Figure 1

Graphical Representation of DFT

(a) Continuous function — Its frequency spectrum

(b) Sampling function — Its frequency spectrum

(c) Time Series — DFT

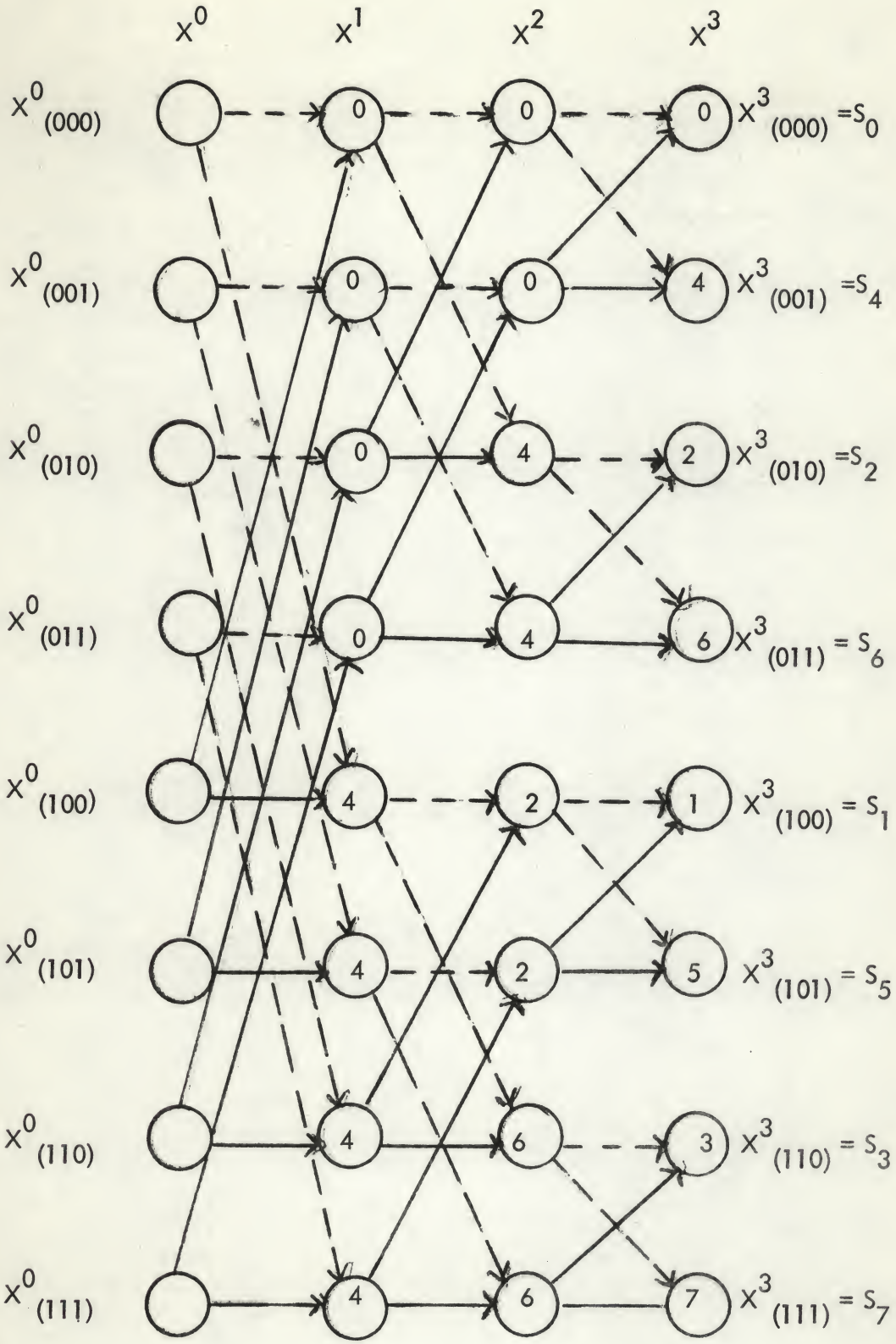


Figure 2

Tree-graph for N=8